

Lehrstuhl für  
Rechnerarchitektur & Parallele Systeme  
Prof. Dr. Martin Schulz  
Dominic Prinz  
Jakob Schäffeler

Lehrstuhl für  
Design Automation  
Prof. Dr.-Ing. Robert Wille  
Stefan Engels

# Einführung in die Rechnerarchitektur

Wintersemester 2024/2025

Übungsblatt 8: RISC-V Prozessor

09.12.2024 – 13.12.2024

## 1 Maschinensprache

- Übersetze das folgende RISC-V Assembly Programm in RISC-V Maschinensprache

```
add s7, s8, s9
sll t1, t2, t3
srli s3, s1, 14
sw s9, 16(t4)
```

168421  
10111

add: R-Type

funct7 | rs2 | rs1 | funct3 | rd | opcode

0000000 | 11001 | 11000 | 000 | 10111 | 0110011

0x 0 1 9 c 0 b b 3

sll: R-Type

funct7 | rs2 | rs1 | funct3 | rd | opcode

0000000 | 1100 | 00111 | 001 | 00110 | 0110011

0x 0 1 c 3 9 7 3 3

srli: I-Type

Immediate 11:0 | rs1 | funct3 | rd | op

0000000 | 1110 | 01001 | 101 | 10011 | 0010011

0 0 e 4 d 9 9 3 = 0x00e4d993

sw: S-Type

Immediate 11:5 | rs2 | rs1 | funct3 | imm 4:0 | op

0000000 | 11001 | 11101 | 010 | 10000 | 0100011

0 1 9 e a 8 2 3 = 0x19e823

0 0000 000 1000 0 = 16

- Konvertiere den folgenden RISC-V Maschinencode zurück in RISC-V Assembly:

0x01200513	0000 0001 0010 0000 0000 0101 0001 0011	addi a0, zero, 18
0x00300593	0000 0000 0011 0000 0000 0101 1001 0011	addi a1, zero, 3
0x00000393	0000 0000 0000 0000 0000 0011 1001 0011	addi t2, zero, 0
0x00058e33	0000 0000 0000 0101 1000 1110 0011 0011	add t3, a1, zero
0x01c54863	0000 0001 1100 0101 0100 1000 0110 0011	blt a0, t3, 16
0x00138393	0000 0000 0000 0000 0000 0000 0000 0000	addi t2, t2, 1
0x00be0e33	0000 0000 1011 1010 0000 1110 0011 0011	add t3, t3, a1
0xff5ff06f	1111 1111 0101 1111 1111 0000 0100 1111	jnl zero, 12
0x00038533	0000 0000 0000 0000 0000 0000 0000 0000	

- Erkläre was das Programm aus der vorherigen Aufgabe berechnet.

1111 1111 0101 1111 1111 0000 0100 1011 1100

addi a0, zero, 18  
 addi a1, zero, 3  
 addi t2, zero, 0  
 add t3, a1, zero  
 loop: blt a0, t3, end  
 addi t2, t2, 1  
 add t3, t3, a1  
 jnl zero, loop  
 end: add a0, t2, zero

a0 a1 t2  
 18 3 0  
 19 3 1  
 20 3 2  
 21 3 3  
 22 3 4  
 23 3 5  
 24 3 6  
 25 3 7  
 26 3 8  
 27 3 9  
 28 3 10  
 29 3 11  
 30 3 12  
 31 3 13  
 32 3 14  
 33 3 15  
 34 3 16  
 35 3 17  
 36 3 18  
 37 3 19  
 38 3 20  
 39 3 21  
 40 3 22  
 41 3 23  
 42 3 24  
 43 3 25  
 44 3 26  
 45 3 27  
 46 3 28  
 47 3 29  
 48 3 30  
 49 3 31  
 50 3 32  
 51 3 33  
 52 3 34  
 53 3 35  
 54 3 36  
 55 3 37  
 56 3 38  
 57 3 39  
 58 3 40  
 59 3 41  
 60 3 42  
 61 3 43  
 62 3 44  
 63 3 45  
 64 3 46  
 65 3 47  
 66 3 48  
 67 3 49  
 68 3 50  
 69 3 51  
 70 3 52  
 71 3 53  
 72 3 54  
 73 3 55  
 74 3 56  
 75 3 57  
 76 3 58  
 77 3 59  
 78 3 60  
 79 3 61  
 80 3 62  
 81 3 63  
 82 3 64  
 83 3 65  
 84 3 66  
 85 3 67  
 86 3 68  
 87 3 69  
 88 3 70  
 89 3 71  
 90 3 72  
 91 3 73  
 92 3 74  
 93 3 75  
 94 3 76  
 95 3 77  
 96 3 78  
 97 3 79  
 98 3 80  
 99 3 81  
 100 3 82  
 101 3 83  
 102 3 84  
 103 3 85  
 104 3 86  
 105 3 87  
 106 3 88  
 107 3 89  
 108 3 90  
 109 3 91  
 110 3 92  
 111 3 93  
 112 3 94  
 113 3 95  
 114 3 96  
 115 3 97  
 116 3 98  
 117 3 99  
 118 3 100  
 119 3 101  
 120 3 102  
 121 3 103  
 122 3 104  
 123 3 105  
 124 3 106  
 125 3 107  
 126 3 108  
 127 3 109  
 128 3 110  
 129 3 111  
 130 3 112  
 131 3 113  
 132 3 114  
 133 3 115  
 134 3 116  
 135 3 117  
 136 3 118  
 137 3 119  
 138 3 120  
 139 3 121  
 140 3 122  
 141 3 123  
 142 3 124  
 143 3 125  
 144 3 126  
 145 3 127  
 146 3 128  
 147 3 129  
 148 3 130  
 149 3 131  
 150 3 132  
 151 3 133  
 152 3 134  
 153 3 135  
 154 3 136  
 155 3 137  
 156 3 138  
 157 3 139  
 158 3 140  
 159 3 141  
 160 3 142  
 161 3 143  
 162 3 144  
 163 3 145  
 164 3 146  
 165 3 147  
 166 3 148  
 167 3 149  
 168 3 150  
 169 3 151  
 170 3 152  
 171 3 153  
 172 3 154  
 173 3 155  
 174 3 156  
 175 3 157  
 176 3 158  
 177 3 159  
 178 3 160  
 179 3 161  
 180 3 162  
 181 3 163  
 182 3 164  
 183 3 165  
 184 3 166  
 185 3 167  
 186 3 168  
 187 3 169  
 188 3 170  
 189 3 171  
 190 3 172  
 191 3 173  
 192 3 174  
 193 3 175  
 194 3 176  
 195 3 177  
 196 3 178  
 197 3 179  
 198 3 180  
 199 3 181  
 200 3 182  
 201 3 183  
 202 3 184  
 203 3 185  
 204 3 186  
 205 3 187  
 206 3 188  
 207 3 189  
 208 3 190  
 209 3 191  
 210 3 192  
 211 3 193  
 212 3 194  
 213 3 195  
 214 3 196  
 215 3 197  
 216 3 198  
 217 3 199  
 218 3 200  
 219 3 201  
 220 3 202  
 221 3 203  
 222 3 204  
 223 3 205  
 224 3 206  
 225 3 207  
 226 3 208  
 227 3 209  
 228 3 210  
 229 3 211  
 230 3 212  
 231 3 213  
 232 3 214  
 233 3 215  
 234 3 216  
 235 3 217  
 236 3 218  
 237 3 219  
 238 3 220  
 239 3 221  
 240 3 222  
 241 3 223  
 242 3 224  
 243 3 225  
 244 3 226  
 245 3 227  
 246 3 228  
 247 3 229  
 248 3 230  
 249 3 231  
 250 3 232  
 251 3 233  
 252 3 234  
 253 3 235  
 254 3 236  
 255 3 237  
 256 3 238  
 257 3 239  
 258 3 240  
 259 3 241  
 260 3 242  
 261 3 243  
 262 3 244  
 263 3 245  
 264 3 246  
 265 3 247  
 266 3 248  
 267 3 249  
 268 3 250  
 269 3 251  
 270 3 252  
 271 3 253  
 272 3 254  
 273 3 255  
 274 3 256  
 275 3 257  
 276 3 258  
 277 3 259  
 278 3 260  
 279 3 261  
 280 3 262  
 281 3 263  
 282 3 264  
 283 3 265  
 284 3 266  
 285 3 267  
 286 3 268  
 287 3 269  
 288 3 270  
 289 3 271  
 290 3 272  
 291 3 273  
 292 3 274  
 293 3 275  
 294 3 276  
 295 3 277  
 296 3 278  
 297 3 279  
 298 3 280  
 299 3 281  
 300 3 282  
 301 3 283  
 302 3 284  
 303 3 285  
 304 3 286  
 305 3 287  
 306 3 288  
 307 3 289  
 308 3 290  
 309 3 291  
 310 3 292  
 311 3 293  
 312 3 294  
 313 3 295  
 314 3 296  
 315 3 297  
 316 3 298  
 317 3 299  
 318 3 300  
 319 3 301  
 320 3 302  
 321 3 303  
 322 3 304  
 323 3 305  
 324 3 306  
 325 3 307  
 326 3 308  
 327 3 309  
 328 3 310  
 329 3 311  
 330 3 312  
 331 3 313  
 332 3 314  
 333 3 315  
 334 3 316  
 335 3 317  
 336 3 318  
 337 3 319  
 338 3 320  
 339 3 321  
 340 3 322  
 341 3 323  
 342 3 324  
 343 3 325  
 344 3 326  
 345 3 327  
 346 3 328  
 347 3 329  
 348 3 330  
 349 3 331  
 350 3 332  
 351 3 333  
 352 3 334  
 353 3 335  
 354 3 336  
 355 3 337  
 356 3 338  
 357 3 339  
 358 3 340  
 359 3 341  
 360 3 342  
 361 3 343  
 362 3 344  
 363 3 345  
 364 3 346  
 365 3 347  
 366 3 348  
 367 3 349  
 368 3 350  
 369 3 351  
 370 3 352  
 371 3 353  
 372 3 354  
 373 3 355  
 374 3 356  
 375 3 357  
 376 3 358  
 377 3 359  
 378 3 360  
 379 3 361  
 380 3 362  
 381 3 363  
 382 3 364  
 383 3 365  
 384 3 366  
 385 3 367  
 386 3 368  
 387 3 369  
 388 3 370  
 389 3 371  
 390 3 372  
 391 3 373  
 392 3 374  
 393 3 375  
 394 3 376  
 395 3 377  
 396 3 378  
 397 3 379  
 398 3 380  
 399 3 381  
 400 3 382  
 401 3 383  
 402 3 384  
 403 3 385  
 404 3 386  
 405 3 387  
 406 3 388  
 407 3 389  
 408 3 390  
 409 3 391  
 410 3 392  
 411 3 393  
 412 3 394  
 413 3 395  
 414 3 396  
 415 3 397  
 416 3 398  
 417 3 399  
 418 3 400  
 419 3 401  
 420 3 402  
 421 3 403  
 422 3 404  
 423 3 405  
 424 3 406  
 425 3 407  
 426 3 408  
 427 3 409  
 428 3 410  
 429 3 411  
 430 3 412  
 431 3 413  
 432 3 414  
 433 3 415  
 434 3 416  
 435 3 417  
 436 3 418  
 437 3 419  
 438 3 420  
 439 3 421  
 440 3 422  
 441 3 423  
 442 3 424  
 443 3 425  
 444 3 426  
 445 3 427  
 446 3 428  
 447 3 429  
 448 3 430  
 449 3 431  
 450 3 432  
 451 3 433  
 452 3 434  
 453 3 435  
 454 3 436  
 455 3 437  
 456 3 438  
 457 3 439  
 458 3 440  
 459 3 441  
 460 3 442  
 461 3 443  
 462 3 444  
 463 3 445  
 464 3 446  
 465 3 447  
 466 3 448  
 467 3 449  
 468 3 450  
 469 3 451  
 470 3 452  
 471 3 453  
 472 3 454  
 473 3 455  
 474 3 456  
 475 3 457  
 476 3 458  
 477 3 459  
 478 3 460  
 479 3 461  
 480 3 462  
 481 3 463  
 482 3 464  
 483 3 465  
 484 3 466  
 485 3 467  
 486 3 468  
 487 3 469  
 488 3 470  
 489 3 471  
 490 3 472  
 491 3 473  
 492 3 474  
 493 3 475  
 494 3 476  
 495 3 477  
 496 3 478  
 497 3 479  
 498 3 480  
 499 3 481  
 500 3 482  
 501 3 483  
 502 3 484  
 503 3 485  
 504 3 486  
 505 3 487  
 506 3 488  
 507 3 489  
 508 3 490  
 509 3 491  
 510 3 492  
 511 3 493  
 512 3 494  
 513 3 495  
 514 3 496  
 515 3 497  
 516 3 498  
 517 3 499  
 518 3 500  
 519 3 501  
 520 3 502  
 521 3 503  
 522 3 504  
 523 3 505  
 524 3 506  
 525 3 507  
 526 3 508  
 527 3 509  
 528 3 510  
 529 3 511  
 530 3 512  
 531 3 513  
 532 3 514  
 533 3 515  
 534 3 516  
 535 3 517  
 536 3 518  
 537 3 519  
 538 3 520  
 539 3 521  
 540 3 522  
 541 3 523  
 542 3 524  
 543 3 525  
 544 3 526  
 545 3 527  
 546 3 528  
 547 3 529  
 548 3 530  
 549 3 531  
 550 3 532  
 551 3 533  
 552 3 534  
 553 3 535  
 554 3 536  
 555 3 537  
 556 3 538  
 557 3 539  
 558 3 540  
 559 3 541  
 560 3 542  
 561 3 543  
 562 3 544  
 563 3 545  
 564 3 546  
 565 3 547  
 566 3 548  
 567 3 549  
 568 3 550  
 569 3 551  
 570 3 552  
 571 3 553  
 572 3 554  
 573 3 555  
 574 3 556  
 575 3 557  
 576 3 558  
 577 3 559  
 578 3 560  
 579 3 561  
 580 3 562  
 581 3 563  
 582 3 564  
 583 3 565  
 584 3 566  
 585 3 567  
 586 3 568  
 587 3 569  
 588 3 570  
 589 3 571  
 590 3 572  
 591 3 573  
 592 3 574  
 593 3 575  
 594 3 576  
 595 3 577  
 596 3 578  
 597 3 579  
 598 3 580  
 599 3 581  
 600 3 582  
 601 3 583  
 602 3 584  
 603 3 585  
 604 3 586  
 605 3 587  
 606 3 588  
 607 3 589  
 608 3 590  
 609 3 591  
 610 3 592  
 611 3 593  
 612 3 594  
 613 3 595  
 614 3 596  
 615 3 597  
 616 3 598  
 617 3 599  
 618 3 600  
 619 3 601  
 620 3 602  
 621 3 603  
 622 3 604  
 623 3 605  
 624 3 606  
 625 3 607  
 626 3 608  
 627 3 609  
 628 3 610  
 629 3 611  
 630 3 612  
 631 3 613  
 632 3 614  
 633 3 615  
 634 3 616  
 635 3 617  
 636 3 618  
 637 3 619  
 638 3 620  
 639 3 621  
 640 3 622  
 641 3 623  
 642 3 624  
 643 3 625  
 644 3 626  
 645 3 627  
 646 3 628  
 647 3 629  
 648 3 630  
 649 3 631  
 650 3 632  
 651 3 633  
 652 3 634  
 653 3 635  
 654 3 636  
 655 3 637  
 656 3 638  
 657 3 639  
 658 3 640  
 659 3 641  
 660 3 642  
 661 3 643  
 662 3 644  
 663 3 645  
 664 3 646  
 665 3 647  
 666 3 648  
 667 3 649  
 668 3 650  
 669 3 651  
 670 3 652  
 671 3 653  
 672 3 654  
 673 3 655  
 674 3 656  
 675 3 657  
 676 3 658  
 677 3 659  
 678 3 660  
 679 3 661  
 680 3 662  
 681 3 663  
 682 3 664  
 683 3 665  
 684 3 666  
 685 3 667  
 686 3 668  
 687 3 669  
 688 3 670  
 689 3 671  
 690 3 672  
 691 3 673  
 692 3 674  
 693 3 675  
 694 3 676  
 695 3 677  
 696 3 678  
 697 3 679  
 698 3 680  
 699 3 681  
 700 3 682  
 701 3 683  
 702 3 684  
 703 3 685  
 704 3 686  
 705 3 687  
 706 3 688  
 707 3 689  
 708 3 690  
 709 3 691  
 710 3 692  
 711 3 693  
 712 3 694  
 713 3 695  
 714 3 696  
 715 3 697  
 716 3 698  
 717 3 699  
 718 3 700  
 719 3 701  
 720 3 702  
 721 3 703  
 722 3 704  
 723 3 705  
 724 3 706  
 725 3 707  
 726 3 708  
 727 3 709  
 728 3 710  
 729 3 711  
 730 3 712  
 731 3 713  
 732 3 714  
 733 3 715  
 734 3 716  
 735 3 717  
 736 3 718  
 737 3 719  
 738 3 720  
 739 3 721  
 740 3 722  
 741 3 723  
 742 3 724  
 743 3 725  
 744 3 726  
 745 3 727  
 746 3 728  
 747 3 729  
 748 3 730  
 749 3 731  
 750 3 732  
 751 3 733  
 752 3 734  
 753 3 735  
 754 3 736  
 755 3 737  
 756 3 738  
 757 3 739  
 758 3 740  
 759 3 741  
 760 3 742  
 761 3 743  
 762 3 744  
 763 3 745  
 764 3 746  
 765 3 747  
 766 3 748  
 767 3 749  
 768 3 750  
 769 3 751  
 770 3 752  
 771 3 753  
 772 3 754  
 773 3 755  
 774 3 756  
 775 3 757  
 776 3 758  
 777 3 759  
 778 3 760  
 779 3 761  
 780 3 762  
 781 3 763  
 782 3 764  
 783 3 765  
 784 3 766  
 785 3 767  
 786 3 768  
 787 3 769  
 788 3 770  
 789 3 771  
 790 3 772  
 791 3 773  
 792 3 774  
 793 3 775  
 794 3 776  
 795 3 777  
 796 3 778  
 797 3 779  
 798 3 780  
 799 3 781  
 800 3 782  
 801 3 783  
 802 3 784  
 803 3 785  
 804 3 786  
 805 3 787  
 806 3 788  
 807 3 789  
 808 3 790  
 809 3 791  
 810 3 792  
 811 3 793  
 812 3 794  
 813 3 795  
 814 3 796  
 815 3 797  
 816 3 798  
 817 3 799  
 818 3 800  
 819 3 801  
 820 3 802  
 821 3 803  
 822 3 804  
 823 3 805  
 824 3 806  
 825 3 807  
 826 3 808  
 827 3 809  
 828 3 810  
 829 3 811  
 830 3 812  
 831 3 813  
 8

## 2 Fehlerhafte Kontrollsignale

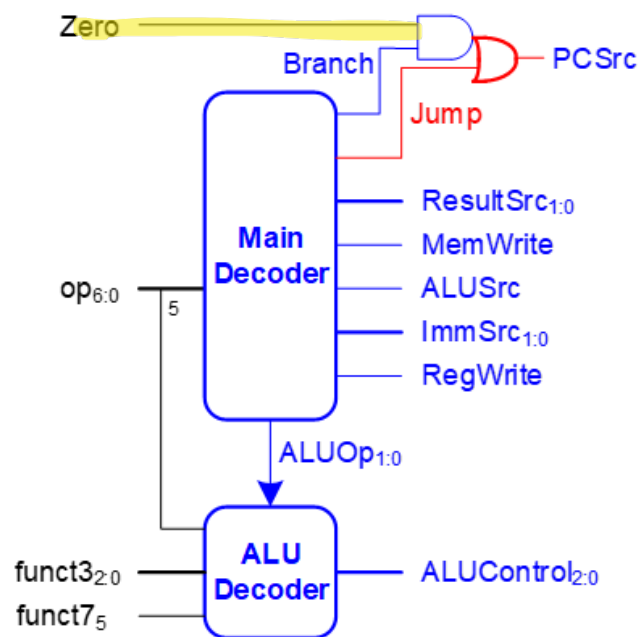
Angenommen der Prozessor in Abbildung 2 hat fehlerhafte Kontrollsignale, deren Wert konstant auf einen bestimmten Wert gelegt ist. Die folgende Tabelle listet Kontrollsignale und deren konstanten Wert auf. Welche der Befehle, die in der Vorlesung vorgestellt wurden, funktionieren nicht mehr unter dieser fehlerhaften Belegung?

Kontrollsignal	Konstante Belegung	Fehlerhafte Befehle
<u>ALUControl</u>	000 (ADD)	R-Typ (außer ADD), I-Typ (außer addi, lw), BEQ
PCSrc	0	BEQ, JAL
ResultSrc	01	R-Typ, I-Typ (außer lw), JAL

© 2024 Lehrstuhl für  
Rechnerarchitektur &  
Parallele Systeme  
alle Rechte vorbehalten

### 3 Erweiterung: BNE

Das Steuerwerk des Prozessors sieht im genaueren wie folgt aus:



Insbesondere ist die Logik des PCSrc Signals genauer abgebildet: PCSrc wird auf 1 gesetzt wenn Jump auf 1 (bei JAL) ist oder ein Branch gemacht wird und das Zero Signal 1 liefert (bei BEQ).

Erweitere das Steuerwerk und gegebenenfalls das Prozessorschaltbild so, dass auch ein BNE (Branch Not Equal) Befehl durchgeführt werden kann. Der BNE Befehl vergleicht zwei Quellregister und addiert einen relativen Offset auf den PC falls die Register *nicht* gleich sind. Der Maschinencode des BNE ist in Abbildung 3 abgebildet.

### 4 Kontrollsignalbelegung

Gebe die Belegung der Kontrollsignale und den Befehlstyp für folgende Befehle an. Für die ALUControl reicht der Name der ALU-Instruktion (ADD, SUB, AND, ...). Gebe Signale bei denen die Belegung keinen Einfluss auf die Funktionalität hat explizit mit *Don't Care* ('x') an.

Befehl	Befehlstype	Branch	ResultSrc	MemWrite	ALUControl	ALUSrc	ImmSrc	RegWrite
OR	R	0	01	0	or	0	xx	1
BNE	B	1	xx	0	sub	0	10	0
XORI	I	0	00	0	xor	1	00	1
SW	S	0	xx	1	add	1	01	0

## 5 Prozessorerweiterung (Hausaufgabe)

Bearbeitung und Abgabe der Hausaufgabe 8 auf <https://artemis.in.tum.de/courses/401> bis **Sonntag, den 15.12.2024, 23:59 Uhr**.

Ziel dieser Übung ist es den in der Vorlesung vorgestellten Prozessor um zwei Befehle zu erweitern und in Digital zu implementieren. Die in der Vorlesung und Zentralübung vorgestellte Funktionalität ist bereits in der Vorlage vorgegeben.

Bevor du mit der Implementierung beginnst, überlege dir zuerst am Schaltbild in Abbildung 2 wie die Kontrollsignale gesetzt werden müssen und welche zusätzliche Komponenten oder Signale eventuell benötigt werden.

Da die Funktionalität des Prozessors getestet werden soll, sind die Testbenches in dieser Übung teilweise Assemblyprogramme. Diese können über Rechtsklick -> **Daten: Bearbeiten** -> **Datei** -> **Laden** direkt ins ROM *InstrMem* in der Datei *SingleCycleRiscV.dig* geladen werden oder selbst geschrieben werden. Auf Artemis sind im Template bereits kleine Programme zum Testen<sup>1</sup> in den *.hex* Dateien vorhanden.

Die folgenden Teilaufgaben können unabhängig voneinander gelöst werden:

- a) Erweitere den gegebenen Prozessor, sodass dieser die Instruktion *xor* unterstützt. In dieser Teilaufgabe dürfen nur die Dateien *ALU.dig* und *ALUDecoder.dig* angepasst werden. Änderungen in anderen Dateien werden vom Tester ignoriert. Die *ALUControl* der *xor*-Funktion soll 100 sein. Die Ein- und Ausgänge dürfen nicht verändert werden.

*Hinweis zum Decoder: Überlege dir zunächst welchen eindeutigen(!) Wert *ALUOp* und *funct3* haben und verbinde eine Konstante an den richtigen Eingang eines Multiplexers.*

- b) Erweitere den gegebenen Prozessor, sodass dieser die Instruktion *jalr* unterstützt. In dieser Teilaufgabe dürfen alle Dateien inklusive des Prozessorschaltbildes geändert werden. Die bestehenden Ein- und Ausgänge dürfen nicht verändert werden und es dürfen keine zusätzlichen Eingänge hinzugefügt werden. Es ist jedoch (nur in Teil b) explizit erlaubt einzelnen Subcircuits zusätzliche Ausgänge hinzuzufügen.

*Hinweis: Die in Teilaufgabe a) angepassten Schaltkreise müssen für diese Teilaufgabe nicht verändert werden. Versuche bei der Implementierung möglichst viele Komponenten wiederzuverwenden.*

*Hinweis:* Es dürfen sämtliche Digital-Komponenten aus den Bereichen „Logisch“, „IO“, „Leitungen“, „Multiplexer“, „FlipFlops“, „Speicher“ und „Arithmetik“ verwendet werden.

---

<sup>1</sup>Um zu sehen ob der Prozessor korrekt ist, überprüfe ob am Ende des Programms die richtigen Ergebnisse im Datenspeicher stehen. Diese werden u.a. im Messwertgraph angezeigt.

## 6 Referenzmaterial

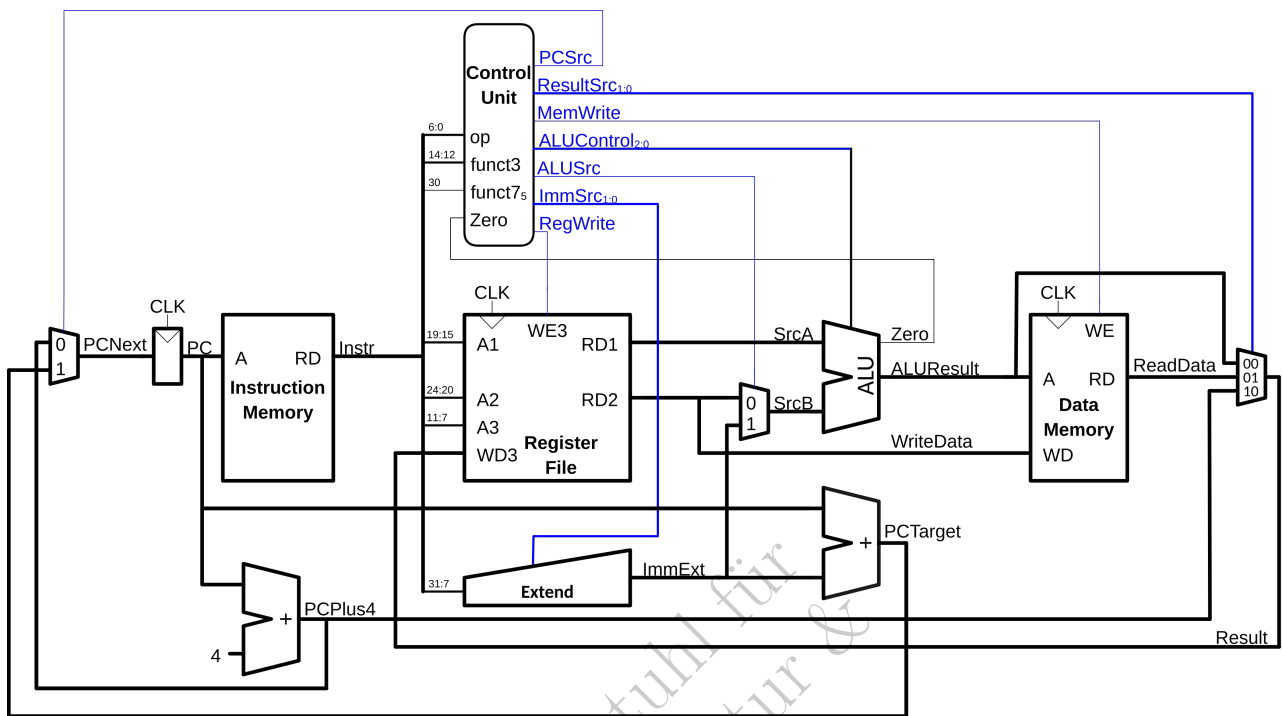


Abbildung 1: Schaltbild des Single Cycle RISC-V Prozessors

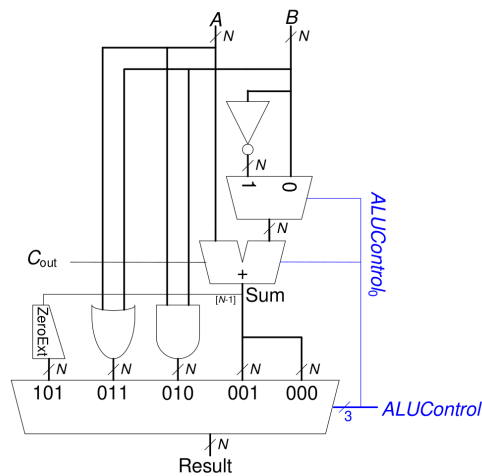


Abbildung 2: ALU des Single Cycle RISC-V Prozessors

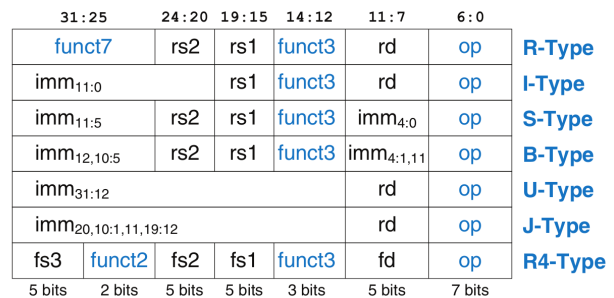
op	funct3	funct7	Type	Instruction	Description	Operation
0000011 (3)	000	–	I	lb rd, imm(rs1)	load byte	rd = SignExt([Address] <sub>7:0</sub> )
0000011 (3)	001	–	I	lh rd, imm(rs1)	load half	rd = SignExt([Address] <sub>15:0</sub> )
0000011 (3)	010	–	I	lw rd, imm(rs1)	load word	rd = [Address] <sub>31:0</sub>
0000011 (3)	100	–	I	lbu rd, imm(rs1)	load byte unsigned	rd = ZeroExt([Address] <sub>7:0</sub> )
0000011 (3)	101	–	I	lhu rd, imm(rs1)	load half unsigned	rd = ZeroExt([Address] <sub>15:0</sub> )
0010011 (19)	000	–	I	addi rd, rs1, imm	add immediate	rd = rs1 + SignExt(imm)
0010011 (19)	001	0000000*	I	slli rd, rs1, uimm	shift left logical immediate	rd = rs1 << uimm
0010011 (19)	010	–	I	slti rd, rs1, imm	set less than immediate	rd = (rs1 < SignExt(imm))
0010011 (19)	011	–	I	sltiu rd, rs1, imm	set less than imm. unsigned	rd = (rs1 < SignExt(imm))
0010011 (19)	100	–	I	xori rd, rs1, imm	xor immediate	rd = rs1 ^ SignExt(imm)
0010011 (19)	101	0000000*	I	srlr rd, rs1, uimm	shift right logical immediate	rd = rs1 >> uimm
0010011 (19)	101	0100000*	I	srair rd, rs1, uimm	shift right arithmetic imm.	rd = rs1 >>> uimm
0010011 (19)	110	–	I	ori rd, rs1, imm	or immediate	rd = rs1   SignExt(imm)
0010011 (19)	111	–	I	andir rd, rs1, imm	and immediate	rd = rs1 & SignExt(imm)
0010111 (23)	–	–	U	auipc rd, upimm	add upper immediate to PC	rd = {upimm, 12'b0} + PC
0100011 (35)	000	–	S	sb rs2, imm(rs1)	store byte	[Address] <sub>7:0</sub> = rs2 <sub>7:0</sub>
0100011 (35)	001	–	S	sh rs2, imm(rs1)	store half	[Address] <sub>15:0</sub> = rs2 <sub>15:0</sub>
0100011 (35)	010	–	S	sw rs2, imm(rs1)	store word	[Address] <sub>31:0</sub> = rs2
0110011 (51)	000	0000000	R	add rd, rs1, rs2	add	rd = rs1 + rs2
0110011 (51)	000	0100000	R	sub rd, rs1, rs2	sub	rd = rs1 – rs2
0110011 (51)	001	0000000	R	sll rd, rs1, rs2	shift left logical	rd = rs1 << rs2 <sub>4:0</sub>
0110011 (51)	010	0000000	R	slt rd, rs1, rs2	set less than	rd = (rs1 < rs2)
0110011 (51)	011	0000000	R	sltu rd, rs1, rs2	set less than unsigned	rd = (rs1 < rs2)
0110011 (51)	100	0000000	R	xor rd, rs1, rs2	xor	rd = rs1 ^ rs2
0110011 (51)	101	0000000	R	srl rd, rs1, rs2	shift right logical	rd = rs1 >> rs2 <sub>4:0</sub>
0110011 (51)	101	0100000	R	sra rd, rs1, rs2	shift right arithmetic	rd = rs1 >>> rs2 <sub>4:0</sub>
0110011 (51)	110	0000000	R	or rd, rs1, rs2	or	rd = rs1   rs2
0110011 (51)	111	0000000	R	and rd, rs1, rs2	and	rd = rs1 & rs2
0110111 (55)	–	–	U	lui rd, upimm	load upper immediate	rd = {upimm, 12'b0}
1100011 (99)	000	–	B	beq rs1, rs2, label	branch if =	if (rs1 == rs2) PC = BTA
1100011 (99)	001	–	B	bne rs1, rs2, label	branch if ≠	if (rs1 ≠ rs2) PC = BTA
1100011 (99)	100	–	B	blt rs1, rs2, label	branch if <	if (rs1 < rs2) PC = BTA
1100011 (99)	101	–	B	bge rs1, rs2, label	branch if ≥	if (rs1 ≥ rs2) PC = BTA
1100011 (99)	110	–	B	bltu rs1, rs2, label	branch if < unsigned	if (rs1 < rs2) PC = BTA
1100011 (99)	111	–	B	bgeu rs1, rs2, label	branch if ≥ unsigned	if (rs1 ≥ rs2) PC = BTA
1100111 (103)	000	–	I	jalc rd, rs1, imm	jump and link register	PC = rs1 + SignExt(imm), rd = PC + 4
1101111 (111)	–	–	J	jal rd, label	jump and link	PC = JTA, rd = PC + 4

\* Encoded in instr<sub>31:25</sub>, the upper seven bits of the immediate field

Abbildung 3: RISC-V 32-bit Integerbefehl

ImmSrc <sub>4:0</sub>	ImmExt	Instr. Type
00	{{20{instr[31]}}, instr[31:20]}	I-Type
01	{{20{instr[31]}}, instr[31:25], instr[11:7]}	S-Type
10	{{19{instr[31]}}, instr[31], instr[7], instr[30:25], instr[11:8], 1'b0}	B-Type
11	{{12{instr[31]}}, instr[19:12], instr[20], instr[30:21], 1'b0}	J-Type

(a) Extend Unit



(b) RISC-V 32-bit Befehlsformat

Name	Nummer	Beschreibung
<b>zero</b>	x0	Konstante 0 (kann nicht überschrieben werden)
<b>ra</b>	x1	Rücksprungadresse
<b>sp</b>	x2	Stack-Zeiger
<b>gp</b>	x3	Zeiger auf globale Daten
<b>tp</b>	x4	Zeiger auf thread-lokale Daten
<b>t0-2</b>	x5 - 7	Temporäre Register*
<b>s0/fp</b>	x8	Stack-Frame-Zeiger (optional)**
<b>s1</b>	x9	Gesicherte Register**
<b>a0-1</b>	x10 - 11	Funktionsargumente/Rückgabewerte
<b>a2-7</b>	x12 - 17	Funktionsargumente
<b>s2-11</b>	x18 - 27	Gesicherte Register**
<b>t3-6</b>	x28 - 31	Temporäre Register*

Abbildung 5: Registertabelle